A Focused Crawler in order to Get Semantic Web Resources (CSR)

Barbosa Santillán, Liliana Ibeth Campos Quirarte, Juana Elizabeth Castro Munguía, Aldo

***Abstract*—This paper presents a Focused Crawler in order to Get Semantic Web Resources (CSR). Structured data web are available in formats such as Extensible Markup Language (XML), Resource Description Framework (RDF) and Ontology Web Language (OWL) that can be used for processing. One of the main challenges for performing a manual search and download semantic web resources is that this task consumes a lot of time. Our research work propose a focused crawler which allow to download these resources automatically and store them on disk in order to have a collection that will be used for data processing.**

CRS consists of three layers: (a) The User Interface Layer, (b) The Focus Crawler Layer and (c) The Base Crawler Layer. CSR uses as a selection policie the Shark-Search method. CSR was conducted with two experiments. The first one starts on December 15 2012 at 7:11 am and ends on December 16 2012 at 4:01 were obtained 448,123,537 bytes of data. The CSR ends by itself after to analyze 80,4375 seeds with an unlimited depth. CSR got 16,576 semantic resources files where the 89 % was RDF, the 10 % was XML and the 1% was OWL. The second one was based on the Web Data Commons work of the Research Group Data and Web Science at the University of Mannheim and the Institute AIFB at the Karlsruhe Institute of Technology. This began at 4:46 am of June 2 2013 and 1:37 am June 9 2013. After 162.51 hours of execution the result was 285,279 semantic resources where predominated the XML resources with 99 % and OWL and RDF with 1 % each one.

I. Introduction

The web is formed by a great collection of resources linked with hypertext or hyperlinks. These resources are created, modified and eliminated continuously. The web as we know it, its designed basically on html which provides textual and graphic information features in order to be understandable by people. This design makes it unfeasible to automatically manipulate information contained on its pages.

A Web Crawler is an agent that searches and downloads web pages automatically. [1] A Focused Crawler is a crawler specialized on some specific subject.[2]

This paper will present the architecture of a Crawler focused on obtaining semantic resources on the web, and creating with them a collection which could be used by another applications or processes.

The remainder of the paper is structured as follows: Section 2 briefly presents the background of our work. Section 3 describes the CSR architecture. Section 4 presents details of our experiments and the result. Finally, Section 5 presents our conclusions and future research.

II. Focused Crawlers

A Web crawler is an algorithm which inspects web pages, methodically and automatically, with the purpose of downloading hyperlinks; and other resources like text, images, videos, etc. from each of this pages. [1]

A crawler stars by listing a group of URLs (Uniform Resource Locator) seeds, which form the queue frontier or waiting list. In this frontier the URLs are stored and prioritized according to the different algorithms. From this queue, and in a predefined order, the crawler obtains an URL, downloads the page, and from this pages it recovers any hyperlink that is contained within its code. These hyperlinks are stored to be part of the frontier. This process can be repeated indefinitely or until the crawler decides to stop according with a parameter: time, quantity, storage space, etc. The downloaded and stored pages form a repository which will be later used by other applications. [1].

The crawlers utilized by general purposes search engines, return an enormous number of web pages independently of the topic. The focused crawlers calculate and assign a higher priority to the pages with greater probability of be relevant according to a specific subject. According to [3] the focused crawlers can be classified as follows:

- Classic focused crawlers: The initial seed group or URLs can be generate by the user as result of a search engine replying an inquiry about the subject [4] [5] [6] [7]. And its then when they lead the search to interesting pages. It utilizes predefined criteria to assign high download priority values to the links, based on the probability of guiding to new pages that refer the topic. The pages with the highest priority are downloaded first [8].
- Semantic crawlers: This is a variant of the focused crawlers. This crawlers use standards of semantic similarity to assign the download priority, which means; a page and a subject can be relevant if they share conceptually (although not necessarily lexical) similar terms. [9]
- Learning crawlers. This crawlers use training process to assign priority to the web pages and guide the crawlers tracing. Usually the learning crawler is given a group of training web pages relevant and non-relevant which are used to accomplish the crawlers learning, assigning high visit priorities to links extracted from web pages classified as relevant to the topic and low priorities to the links extracted from non-relevant pages.[6]

The main problems faced by Web Crawlers are:

- The webs size. At a given time, the crawler can only access a small portion of the web; hence it is essential to give priority in order to select the most important URLs and visit them.
- The speed at which the web changes. It is necessary to have a parameter that marks the frequency at which the pages must be revisited since there is a high probability that when the download of a group of URLs is finished, the visited pages have already been changed, eliminated or pages of mayor interest have been generated.
- Dynamism on the web. The content of most of the web pages is generated in a dynamic manner, that is to say, according with the specific requests of the user after taking forms or surveys designed to be answered by humans, which makes access to said pages or contents hard for an automatic program.

In order to counteract this problems the crawlers use different alternatives or models which allow to calculated the: policies of selection (Rank the pages of the frontier to choose which URLs to visit), Policies of review (select how often to update the data collection, and how), Policies of courtesy (calculate the parameters of courtesy in order to avoid overloading the servers such as waiting time limit, maximum time of connected, bandwidth usage, etc.) and policies of parallelization (manages and marks the URLs of the frontier so it prevents two different process from visiting the same page)

Utilizing selection policies, the crawlers try to lead the process of tracing pages relevant to the topic. [10] [11] The efficacy of the crawler depends of the good selection of seed pages (initial pages). Good seed pages can be pages relevant to the subject of pages with access to the pages of interest with a small number of router hops. For example, if the subject is children books, a good seed page could be the page of the Publisher of an author of children books. But it could also be the authors personal page; that although it could not have any publications, it could guide to pages that contain the wished results.

Some of the most used selection policies are:

- Breath First. This scheme was utilized by the first crawlers, it consists on tracing a domain starting from the root, downloading all the documents on that depth level, before downloading any link on another level. In other words, it starts by visiting the home page of every seed web sites and the new pages are added at the end of the list [1].
- Depth First. Consists on tracing a domain starting from the root or seed, downloading first all the documents available through a URL in particular, before moving on to another link on the list [12].
- Fish-search focus [13] assigns binary values of priority (1 for relevant, 0 for non-relevant) to the pages candidate to be downloaded trough a simple comparison of words. Therefore, all the relevant pages have the same priority value .
- Best-First Crawlers assigns priority values to the candidate pages calculating the similarity of its text with the subject and valuating the frequency of occurrence, applying VSM (Vector Space Model). It does not utilize the inverse indexing since this is problematic due not only to the necessity of calculating the vectors on each step of the tracing but also, on the first stages of the process, the values are very inaccurate due to the number of documents being to small. Best-First is considered the most successful approach to focused tracing due to its simplicity and efficiency [14].
- Backlink-count. This scheme consists on tracing first the pages with the highest number of links directing to it, this way the next page to be analyzed is the one most directed to from the previously downloaded pages [15].
- PageRank. This metric can only be utilized when the user already have a subset of pages available in the collection or in the revisits. It consists on defining recursively the importance of a page as the weighted sum of the previous pages with links to it[1].
- The Shark-Search method [16] suggests using the Vector Space Model(VSM) [17] to assign non-binary priority values to the candidate pages. The priority values are calculated taking on consideration the content of the page, hyperlinks, text surrounding the links and the priority value of the parent pages (pages that direct to the page that contains the link).
- The N-Best-First crawling [8] its a generalized version of the Best-First crawling: on each step, the N-pages (instead of just one) with the highest priority are selected for the expansion.
- The Smart Tracing [18] suggest to combine the content of the page, information of the URL chain, sibling pages and statistics about the relevant and non-relevant pages to assign priorities to the candidate pages. This results on a highly effective tracing algorithm that learns to trace without direct training [1].

III. ARCHITECTURE

This chapter will explain the architecture and the functions of the crawler and each component. CSR crawler is based in a Open source Project that provides a quickly structure to explore the Web, which allows to execute multiple threaths to reduce the time of searching.

The most processing of the crawler is in the engine which is composed by two layers (CSR Layer and Crawler layer), is intented to get as many data file as the user determined by the number of seeds.

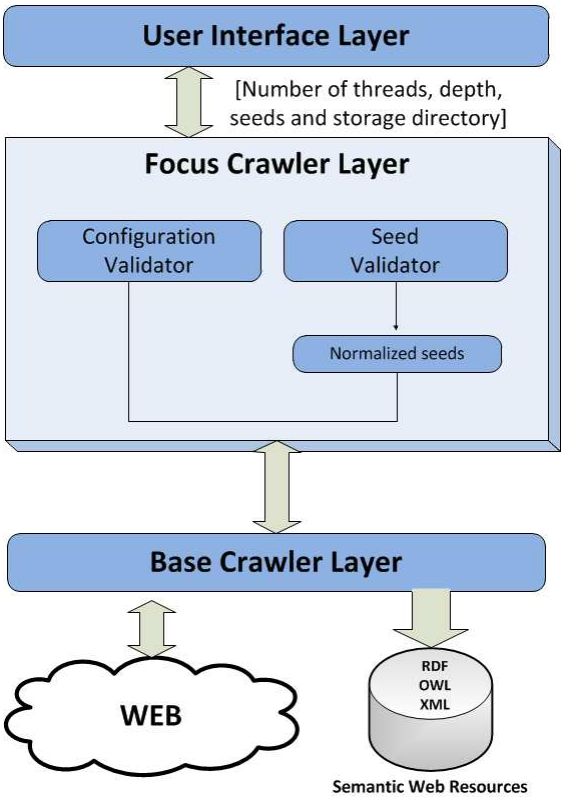


Fig. 1. Architectur of CSR

The storage directory is the specific path to the storage unit on the hard disk; this path will gather all the documents obtained by the crawler during the task.

The threads are instances of a process, which perform a specific task, in this case the search for semantic resources, which reduces the time significantly. The user must try to not overload the system and cause and unexpected termination of the crawler.

The depth is the link levels that a seed can have, then if a seed has another link embedded in the body of the web this will conduct a search on this link, and so on up to the number of levels configured on the system. It is important to note that in unlimited number of levels can take months of processing in multiple threads.

The seeds will be brought to the engine by a simple text file, which will be analyzed and manipulated by the engine, this is the point of departure for the CSR, once the validation is complete will be added to the list to be crawled. The system is divided into three layers: User Interface, Focus Crawler and Basic Crawler as shown in Fig. 1.

A. *User Interface Layer*

It is the interaction to the user, who can enter the data required to specify the seeds, the number of threads and the depth of the crawling, it is important to note that once the system is configurated there is no relevant information that the user may need, when the system complets the crawling it will display a message generated by the following layers and the process will end.

The interaction between this layer and the next layer will be made by sending the required as parameters, if any of these are missing the process can not continue.

B. *Focus Crawler Layer*

Is where the most of the development of this work is done, containing the seed handling and the subsystem to check the already searched pages.

The validator is the first thing that will be run with the input data, once the data is validated, it start storing the seeds in a temporaly program variable, which has a mximum of 300 thousand seeds per execution, this is to avoid overloading the processing and not overflow the memory computer, this process will only check the right structure in the seed.

The inspection process on the web takes a lot of processing, seed files can take days, weeks or months to be procesed, its really important that the crawler avoid check for second time a seed that was already checked, all the pages insepected are going to be stored in a temporaly file.

The controller is the part that handles the logic of the crawler, it will use all the packages and utilities provided by the Basic Crawler Layer, initialize, review and stop the execution threads, it is the brain for the operation .

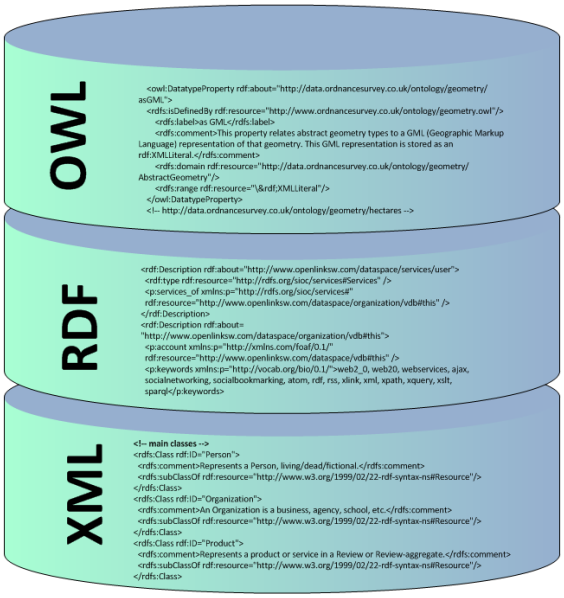
The filters are going to be used to select those files and documents that have relevant data for further analisys, if the data doesnt containt important data will be ignored. Each data structure has its own algoritm to filter and will interact with the lowest layer of the crawler.

C. *Basic Crawler Layer*

Last layer in the Project, created by Yasser Ginjisaffar in the 2008, will provide an API to the Focus Crawler layer to crawl the web [**?**].

IV. Experiment and Results

In order to obtain semantic resources we have specified three data structure which are the resources that the CSR crawler searched:



The main objective of this experiment was to obtain as many semantic resource files from a seed file. RDF, XML and OWL files are considered semantic resources files, wich contains relative information about the data, at the end of the experiment we expect to have a database containing semantic resources files wich can be analyzed for future use. An example of each file is show above, the structure of the files are very similiar.

A. *Experiments*

The main objective of this experiment was to obtain as many semantic resource files from a seed file. There are important variables that need to be measured to determine the importance of our experiment, these are how many seeds are inserted, how many are crawled, how many respond an error, how many threads are used , the deepth of the crawling, downloaded files, total files and their space on disk in a specific time. For both experiment we use the configuration as shown in Table I:

Parameter	Value	Comment
Resume	False	Resume a previously crashed crawl
Depth	Unlimited	Maximum depth of crawling
Robotstxt	True	Should the crawler obey the Robots.txt
Threads	5	Max running threads
Format	RDF/XML/OWL	Default ontology format
Socket timeout	20000	Socket timeout in miliseconds
Connection timeout	30000	Connection timeout in miliseconds
Max download size	1048576	Max size of a file is 1Mb
User agent	Crawler4j	User agent
Politeness delay	200ms	Time to crawl other page in the same server
Follow redirects	True	Should the crawler follow redirects?

The Focus CSR crawler uses multi-thread to download data of web, visit, calculate and filter at the same time. There are 5 threads working in the first execution.

1) First Experiment: The first experiment starts on December 15 2012 at 7:11am and ends on December 16 2012 at 4:01 am, in this we get 426.34 Mbs (448,123,537 Bytes) of data. The CSR crawler finish by itself after analyze an approximate of 80,4375 seeds whit an unlimited depth.

For this experiment the seeds were randomly selected from a database of previously generated, not perform any prioritization of the seeds. These are someof the results:

```
Example of the seed file:
<http://www.johngoodwin.me.uk/family/I0377>
<http://purl.org/vocab/relationship/grandparentOf>
<http://www.johngoodwin.me.uk/family/I0456>
<http://www.johngoodwin.me.uk/family/I0377>
<http://www.johngoodwin.me.uk/family/I2566>
<http://www.johngoodwin.me.uk/family/I0377>
<http://www.kanzaki.com/ns/music\#Musical\_Work>
<http://www.w3.org/2002/07/owl\#Class>
<http://www.kanzaki.com/ns/music>
<http://statistics.data.gov.uk/id/country/921>
<http://statistics.data.gov.uk/def/electoral-
geography/ward>
<http://statistics.data.gov.uk/id/electoral-
ward/18UHGW>
<http://statistics.data.gov.uk/def/electoral-
geography/ward>
<http://statistics.data.gov.uk/id/electoral-
ward/00MENX>
<http://statistics.data.gov.uk/doc/country/921>
```

This kind of seeds were introduced into the CSR crawler. After crawls 22.5 hours we save 16,576 semantic resources files, in these case the Resources Data Framework file was de predominance file with the 89 percent of the files obtained (14,825) leaving the XML file as second place with 10 percent (1736) and finally the Ontology Web Language resource with one percent (15 files).

Files:

XML Files	RDF Files	OWL Files	Summary
1736	14825	15	16576

Data:

XML Files	RDF Files	OWL Files	Summary
152MB	2735MB	1.34MB	426.3 MB

2) Second experiment: The second experiment was based on the work of Web Data Commons, created by the Research Group Data and Web Science at the University of Mannheim and the Institute AIFB at the Karlsruhe Institute of Technology this project extracts all the embedded structured data describing products, people, organizations, places or events into their HTML pages from several billion web pages. Unlike this project, we are looking for the entire file of semantic resoruces, Web Data Common were looking for specific data embedded in a HTML document

This project provides us whit a huge base of seeds that we can use to crawl the web, this information is compress into N-Quads and it can be downloaded from their official server. There are 410 N-Quad files with a size approximately of 100 MBytes each. Uncompressed this kind of file could reach 1.5 Gbs with 9,278,514,430 seeds to crawl. In this case was necessary to standardize the nq file to use them in our seeds protocol as shown in example .

Example 1 of the NQ file from WebData commons project before normalization.

```
<http://turcanu.net/blog/2008/07/16/honglaowai-
if-there-were-no-communist-party-then-there-
would-be-no-new-china/>
<http://creativecommons.org/ns#attributionURL>
<http://turcanu.net>
<http://turcanu.net/blog/2008/07/16/honglaowai-
if-there-were-no-communist-party-then-there-
would-be-no-new-china/>
<http://turcanu.net/blog/2008/07/16/honglaowai-
if-there-were-no-communist-party-then-there-
would-be-no-new-china/>
<http://creativecommons.org/ns#attributionName>
"Sergiu Turcanu"
<http://turcanu.net/blog/2008/07/16/honglaowai-
if-there-were-no-communist-party-then-there-
```

```
would-be-no-new-china/>
<http://www.telemac0.net/marketing-50/>
```

Example 2 of the NQ file from WebData commons project after normalization.

```
"http://turcanu.net/blog/2008/07/16/honglaowai-
if-there-were-no-communist-party-then-there-
would-be-no-new-china/"
"http://creativecommons.org/ns#attributionURL"
"http://turcanu.net"
"http://turcanu.net/blog/2008/07/16/honglaowai-
if-there-were-no-communist-party-then-there-
would-be-no-new-china/"
"http://turcanu.net/blog/2008/07/16/honglaowai-
if-there-were-no-communist-party-then-there-
would-be-no-new-china/"
"http://creativecommons.org/ns#attributionName"
"http://turcanu.net/blog/2008/07/16/honglaowai-
if-there-were-no-communist-party-then-there-
would-be-no-new-china/"
"http://www.telemac0.net/marketing-50/"
```

After this file was normalized to be an input of the CSR crawler, the CSR crawler was set with the default configuration and start crawling, this experiment was realized between 04:46AM of 02 June 2013 and 01:37AM of 09 June 2013, the project crawl 162.51 hours, the results of these crawling session were 285,279 of semantic resources retreive from the Web In this case the predominance file was the Extensible Markup Language with the 99 percent of the saved files (285279 XML files), leaving RDF files and OWL with .1 percent each.

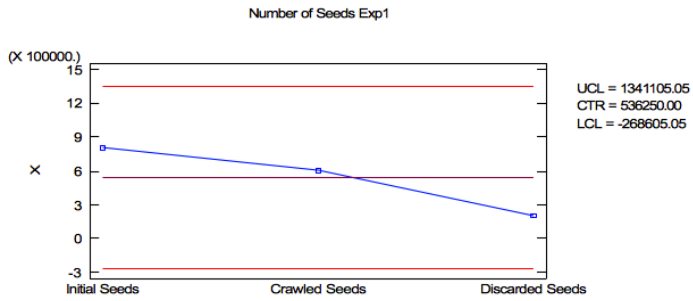
In this experiment more that 60 percent of the seeds files weren't crawled, this may be because the page was moved, removed or the robots policy donot allow to crawl this URL.
Results:

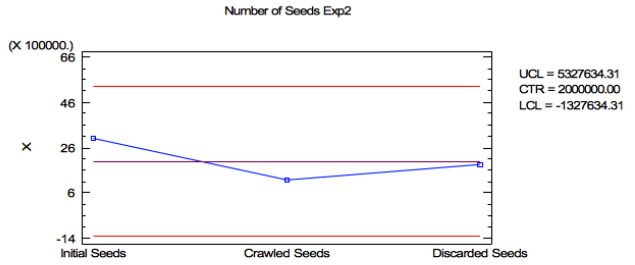
Files:

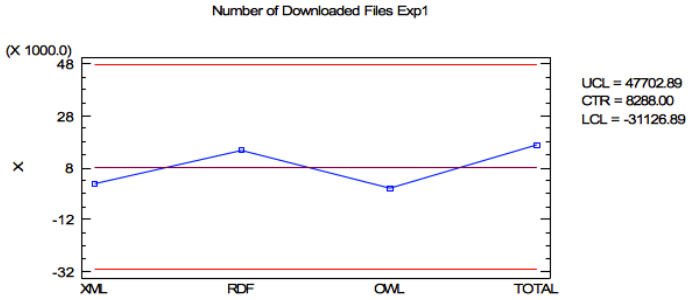
XML Files	RDF Files	OWL Files	Summary
284922	325	32	285279

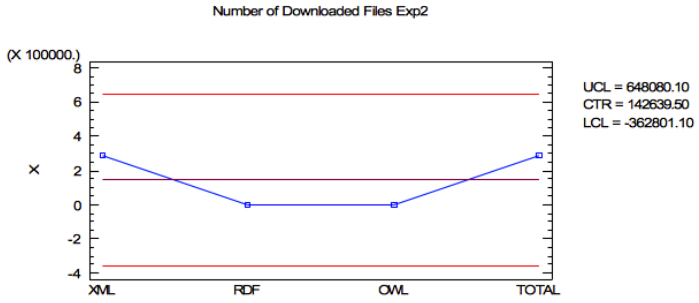
Data:

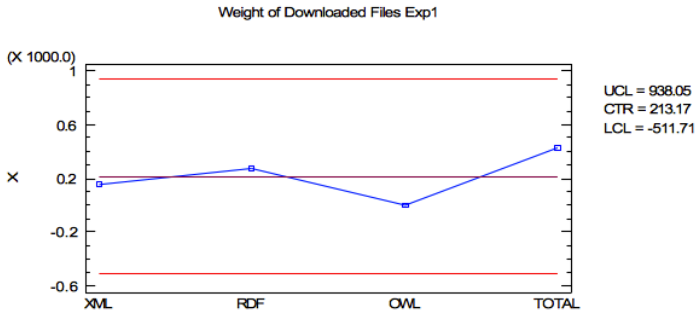
XML Files	RDF Files	OWL Files	Summary
3150MB	19.5MB	5.1MB	3175 MB/3.1 GB











V. Conclusion

The semantic web resources are growing nevertheless not all the web sites use them. The results of CSR showed that the XML and RDF resources are more frequently In contrast to OWL that does not have much presence. However with the update of the web sites to the semantic Web they will use this kind of resources.

CSR used a X policie for focused crawlers for the two experiments. The most important part was in the cleaning and normalization of the seeds. On the one hand was with open

seeds and and on the other with seeds provided by the project of Web Data Commons del Research Group Data and Web Science at the University of Mannheim and the Institute AIFB at the Karlsruhe Institute of Technology. Future work is to parallel the CSR tasks in order to obtain as many semantic resources as possible in a period of time for a later process. Another major challenge is automatically check the quality of the results.

ACKNOWLEDGMENT

The authors would like to thank PROMEP for funding this research project.

REFERENCES

[1] J. Cho and H. Garcia-Molina, "Parallel crawlers," pp. 124–135, 2002.

[2] A. A. Barfourosh, H. M. Nezhad, M. O'Donovan-Anderson, M. Odonovan-Anderson, A. Kabir, and D. Perlis, "Alii: An information integration environment based on the active logic framework," 2002.

[3] S. Batsakis, E. G. M. Petrakis, and E. Milios, "Improving the performance of focused web crawlers," *Data Knowl. Eng.*, vol. 68, no. 10, pp. 1001–1013, 2009.

[4] F. McCown and M. L. Nelson, "Agreeing to disagree: search engines and their public interfaces," pp. 309–318, 2007.

[5] R. Kraft and R. Stata, "Finding buying guides with a web carnivore," pp. 84–, 2003.

[6] G. Pant and P. Srinivasan, "Learning to crawl: Comparing classification schemes," *ACM Trans. Inf. Syst.*, vol. 23, no. 4, pp. 430–462, 2005.

[7] S. Bao, R. Li, Y. Yu, and Y. Cao, "Competitor mining with the web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 10, pp. 1297–1310, 2008.

[8] F. Menczer, G. Pant, and P. Srinivasan, "Topical web crawlers: Evaluating adaptive algorithms," *ACM Trans. Internet Technol.*, vol. 4, no. 4, pp. 378–419, Nov. 2004.

[9] M. Ehrig and A. Maedche, "Ontology-focused crawling of web documents," pp. 1174–1178, 2003.

[10] A. Pivk, P. Cimiano, Y. Sure, M. Gams, V. Rajkovič, and R. Studer, "Transforming arbitrary tables into logical form with tartar," *Data Knowl. Eng.*, vol. 60, no. 3, pp. 567–595, 2007.

[11] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan, "A survey of web information extraction systems," *IEEE Trans. on Knowl. and Data Eng.*, vol. 18, no. 10, pp. 1411–1428, Oct. 2006.

[12] H. Ali, "Effective web crawlers," March 2008.

[13] P. D. Bra, G.-J. Houben, Y. Kornatzky, and R. Post, "Information retrieval in distributed hypertexts." pp. 481–493, 1994.

[14] J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through url ordering," *Comput. Netw. ISDN Syst.*, vol. 30, no. 1-7, pp. 161–172, 1998.

[15] C. Castillo, M. Marin, A. Rodriguez, and R. Baeza-Yates, "Scheduling algorithms for web crawling," pp. 10–17, 2004.

[16] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur, "The shark-search algorithm. an application: tailored web site mapping," *Comput. Netw. ISDN Syst.*, vol. 30, no. 1-7, pp. 317–326, 1998.

[17] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975.

[18] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu, "Intelligent crawling on the world wide web with arbitrary predicates," pp. 96–105, 2001.